An Integer Based Hierarchical Representation for VLSI

Telle Whitney† and Carver Mead
California Institute of Technology
Pasadena, California 91125

1 Introduction

Geometries with 45° line segments are often used in integrated circuit layouts, since they can save considerable area. In the limit, the introduction of 45° lines is only 5% less dense than optimal geometry, i.e. circular geometry, whereas manhattan geometry is 27% less dense [7]. Obviously any actual design cannot make use of this density factor everywhere — but figure 1 illustrates a simple but common routing problem where the introduction of 45° wires substantially reduces the area. There has been a trend towards strict manhattan geometries in recent years, however, since it is commonly believed that design rule checking is complicated by the inclusion of intermediate angles [11, 1, 3, 6]. This paper describes a hierarchical representation that supports a complete circuit description, but restricts the set of allowable lines to be horizontal, vertical and 45°. Points are constrained to lie on an integer grid. Rather than use arbitrary polygons, transistors and connection wires are constructed from *paths* whose sides and ends are created from an octagonal *circle approximation*. The geometry for contacts is octagonal and is generated from the same circle approximation. The integer grid and restricted line styles allow the simplification of all the Geometrical Design Rule (GDR) checking algorithms — for example a square root is not required in the point-point distance calculations, and division is never required. In fact this approach requires less computation than typical manhattan systems. All of the calculations that normally require real number representation are expressed in integers, eliminating any possibility of round-off errors.

An integer grid provides the basis of the circuit description where every point on the unit octagon falls on the grid. Since the geometry for the transistors, interconnection wires and contacts is generated based on this unit octagon, all the primitive elements lie completely on the grid. Since transistors are represented explicitly, the circuit connectivity is easily maintained; i.e., the representation is self extracting. The effective use of 45° lines not only simplifies the circuit description but reduces the design area without any computational cost.

The methodology presented in this paper guarantees that a composition of two or more circuit descriptions is correct, given that the individual circuits are correct.

---

† currently at Schlumberger Palo Alto Research Center

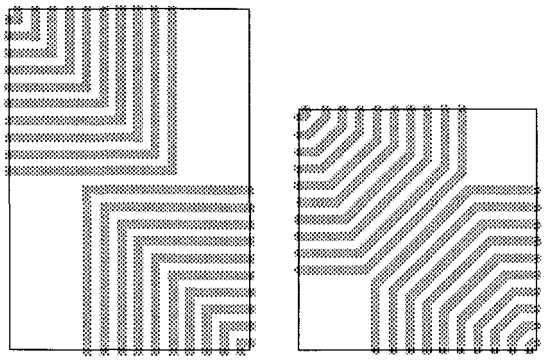


Figure 1: Routing With 45° Wires

In this paper, the term *correct* means that the design is both GDR correct, and that the circuit connectivity is correctly maintained. A cell abstraction capability facilitates and supports hierarchical design. A *cell* is a self contained circuit description. Each cell is fully represented during the composition process by a cell interface with sufficient information to produce a correct composition with another cell. The interface consists of a set of sides that form a simply connected polygon that bounds the cell, and a complete set of the primitive elements that occur at the cell boundary. Each side includes a list of ports that indicate how this cell communicates with other cells. Ports must lie on the integer grid, and each side is chosen from the valid line styles.

A system that supports this methodology has been developed and successfully applied to both nMOS and cMOS bulk technologies. The notion of a path was easily extended to represent cMOS well regions with the first and last points of the well paths equivalent.

2 Pooh Overview

The Pooh representation [9, 10], developed at Caltech, is a circuit representation for geometry with arbitrary angles. It provides an environment in which to explore the necessary characteristics of a design system without introducing dependence on a particular approximation. Circles, lines, and arcs are general geometrical objects, and form the basic elements from which transistors, wires and connection points are formed. The Pooh study has provided valuable insight into the crucial problems in circuit and layout design.

After exploring the necessary components of a general approach to a problem, it is always useful to look for simplifications that do not detract from the elegance of the original solution. A system that supports the complete representation is a useful tool for some applications, but introduces unnecessary computational inefficiencies under many circumstances. This section presents an overview of the general Pooh representation described in detail in [10]. The next section describes how to simplify the computations without losing the elegance.

The problem addressed by the Pooh representation is how to hierarchically define and interconnect transistors in a strictly legal way. The primitive elements in the

representation are lines, arcs and circles. Transistors, interconnection wires, and connection points are defined in terms of these elements. There are three types of algorithms necessary to ensure the correct description of the topology, geometry, and connectivity:

- Synthesis algorithms that allow the construction of GDR correct transistors, wires and connections points.
- Node propagation algorithms that maintain the circuit node information.
- Analysis algorithms that check the interactions between the elements.

The composition of cells is key to making any representation a viable description for VLSI. First, an abstraction of the cell is derived. Then cells are composed to form the next level in the hierarchy. In Pooh, a legal composition guarantees the topology, geometry and connectivity of the composite structure.

Primitive elements are defined in terms of paths and points, rather than conventional polygons and boxes. Each path, as shown in figure 2, is composed of a sequence of segments where each segment consists of a line segment and an arc. The line segment is directional, and is represented by its normalized line equation $L_i(x,y) = A_i x + B_i y + C_i$, where $A_i^2 + B_i^2 = 1$. The arc is represented by the end point of the previous line segment ep_i, the starting point of the next line segment bp_{i+1}, and a signed arc radius r_i, as shown in the figure. A positive radius indicates a counter-clockwise direction, and a negative radius indicates a clockwise direction. If $r_i = 0$ then the arc collapses to a point. Circles are the means of representing the points and generating the arcs, and coupled with the line equations provide the basis for the operational algorithm simplicity.

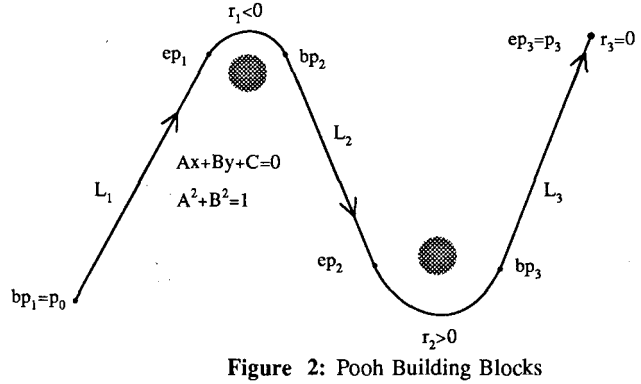


Figure 2: Pooh Building Blocks

There are two types of algorithms that allow the creation of strictly legal leaf cells. The synthesis algorithms, described fully in [10], allow each primitive element to be constructed according to the GDR correctness criteria contained in a technology file. These algorithms depend extensively on the line and arc representation and the line intersection equation. The point p_i where the two lines $L_1(x,y)$ and $L_2(x,y)$ intersect is:

$$\begin{aligned} \det &= B_1A_2 - A_1B_2 \\ p_i &= \big((B_2C_1 - B_1C_2)/\det,\ (A_1C_2 - A_2C_1)/\det\big). \end{aligned} \tag{1}$$

The more useful simplification used for construction purposes between a line $L_1(x, y)$ and its perpendicular $L_\perp(x, y)$ is:

$$\begin{aligned} L_\perp(x, y) &= -B_1x + A_1y + C_2 \\ \det &= -1 \\ p_i &= (B_1C_2 - A_1C_1, -A_1C_2 - B_1C_1). \end{aligned}$$

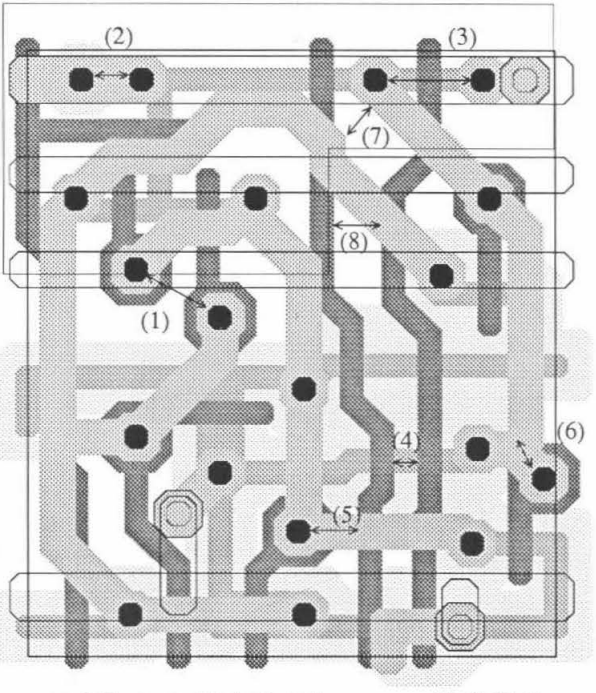
The analysis algorithms, illustrated in figure 3, allow the detection of violations between elements that are spatially "close". Because most of the difficult GDR constraints are met during the synthesis phase, the remaining comparisons during analysis are simple spacing and angle checks, described fully in [10]. The comparisons are between point-line, point-point, arc-line, and arc-arc, and use the following simple distance calculations:

$$PointPoint(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{2}$$

$$LinePoint(L_i, p_q) = A_ix_q + B_iy_q + C_i. \tag{3}$$

Given a GDR correct cell, the cell interface is defined by a set of sides $\mathcal{Q}$. The set of sides $Q_i \in \mathcal{Q}$ bounds all points within the cell and forms a clockwise simply connected polygon. Each side Q_i is defined by the normalized line equation $L_i(x, y)$, described earlier, and contains a sorted list of ports that indicate the cell's external connection points.

For each side Q_i, whose placement is defined by $L_i(x, y)$, it is possible to construct a line $L_i'(x, y)$ closer to the cell center by a distance D, where $A_i' = A_i$, $B_i' = B_i$, and $C_i' = C_i + D$. This line equation, in conjunction with the line intersection equation (1), may be used to derive a set of path segments and points that fall within a distance D of the cell boundary $\mathcal{Q}$. In fact, D varies for each path segment and point, and is set to the maximum applicable design rule for each element. A rigorous treatment of this derivation is given in [10]. Figure 4 illustrates the composition process for a cMOS bit serial multiplier [4]. A composition between two cells is formed by superimposing two sets of ports — one from each cell, and then merging the two port sets, as shown in figure 4(b). Assuming that the cell definition ports include a node number that indicates the connectivity within the cell, it is possible to derive the connectivity for each cell instance, and guarantee the GDR correctness for each level in the hierarchy, maintaining a minimal amount of information in the process. Although the boundary in figure 4 uses strictly orthogonal sides, Pooh does not restrict the sides to be either orthogonal or limited in number. Instead, the boundary may be any simply connected polygon where each side is defined by a valid line equation.



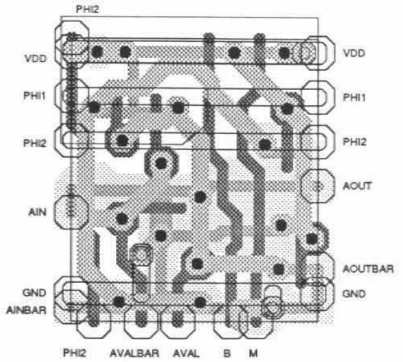
(1) Unconnected Contacts	[*PointPoint*]	**Path-Path**	
(2) Connected Contacts	[*PointPoint*]	(6) Path Angle	
(3) Transistor Overlap	[*PointPoint*]	(7) Line Segment-Point	[*LinePoint*]
(4) Connected Transistors	[*PointPoint*]	(8) Line Segment-Arc	[*LinePoint*]
(5) Contact to Path	[*LinePoint*]	(9) Arc-Arc (*not shown*)	[*PointPoint*]

Figure 3: Analysis Checks

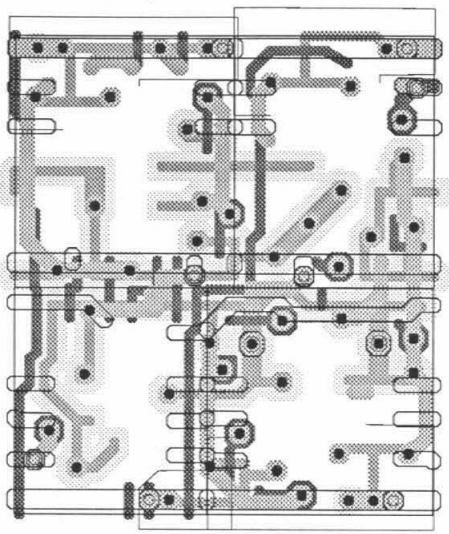
3 Simplified Representation

The simplified approach described in this paper uses an octagon circle approximation to support vertical, horizontal and 45° lines. Using only these line styles, Pooh arcs easily map into a sequence of connected line segments, as shown in Figure 5. These path sequences are often conveniently notated by the designer as an arc; however, the arc in the representation is an unnecessary complication for the relatively small sequence of segments encountered with the line styles used with this approximation.

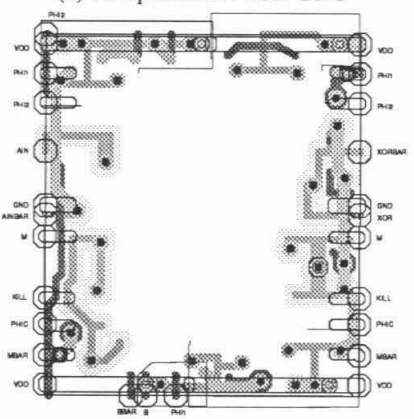
Cell sides are restricted to be one of the three valid line styles. The points of the unit octagon may be represented as integers, and all of the basic Pooh calculations are both simplified and integer based, with obvious implications for efficiency. This technique for simplification may be applied using other circle approximations. For example, an alternative approach is to use a hexagon as the circle approximation and restrict the line styles to orthogonal, 30°, and 60°. Alternatively, one could use a box as the circle approximation and restrict the lines



(a) Original Cell Definition



(b) Composition of Four Cells



(c) New Cell Abstraction

Figure 4: Cell Abstraction for One Bit of a Serial Multiplier

to vertical and horizontal and have a manhattan system, but there is little point since no computation is saved. Designers typically prefer 45° lines, but we are currently investigating some designs that map nicely into hexagonal geometries.

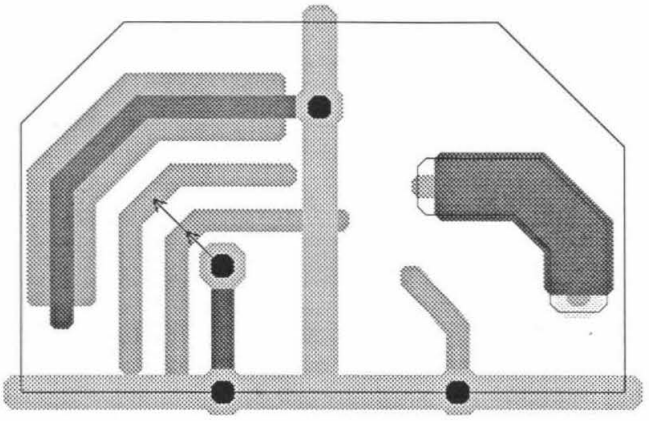


Figure 5: Example nMOS Transistors, Wires, Contacts and Sides

4 The Circle and the Grid

The restricted methodology relies on an integer grid, where the coordinates of valid grid points are even integers. Figure 6 illustrates the integer grid. Valid grid points are designated by the small dotted circles in the figure. One design layout unit (λ for example) is equivalent to four grid points. Half λ is two grid points, and is the smallest increment between the coordinates of valid addressable points. Points with odd coordinates are forbidden under all circumstances in order to prevent non-integer values as intermediate results.

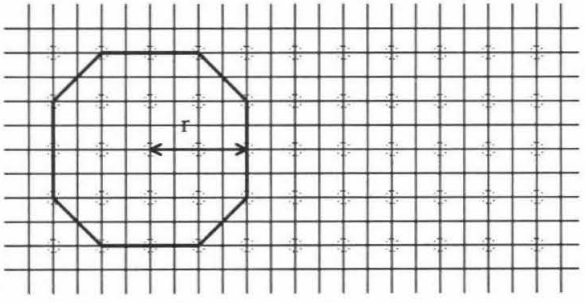


Figure 6: The integer grid and Unit Circle

The underlying unit circle approximation is an octagon of radius four, as shown in Figure 6. Using this circle as the basic unit of measure, the range of the integer system is sufficient in λ units [2], in microns, or in CIF units (hundredth of a

micron). For example, on a 32 bit computer, with one layout unit equal to a hundredth of a micron, the largest representable unit, or every fourth grid point, is 2^{30}. For signed numbers, the range of units is $\pm 2^{29}$. Typical chips today are a centimeter on a side, or 10^4 microns. Wafers are approximately $10 - 20$ centimeters on a side, or 10^7 hundredths of a micron. Thus any design conceivable today is easily representable in this numbering scheme, with plenty of resolution remaining for devices of ultimately small dimension.

This particular octagon approximation was chosen primarily because its end points are on a simple even grid, and the lines generated based on this circle approximation are on the same grid. Although this octagon is not truly circular, its deviation from an actual circle is very tolerable, as illustrated in Figure 7.

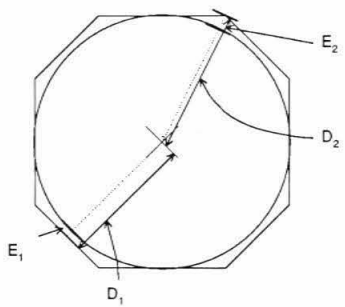


Figure 7: Octagon Error

The distance from the oblique edges of the octagon to the octagon center is $D_1 = \sqrt{18} = 3\sqrt{2} \approx 4.24$. The deviation from the circle radius is $D_1 - 4$, or .24, with a relative error of $E_1 = .24/4 = .06$. The distance from the octagon corners to the center is $D_2 = \sqrt{20} = 4\sqrt{5} \approx 4.47$. The deviation from the circle radius is $D_2 - 4$, or .47, or a relative error of $E_2 = .47/4 = .118$.

5 Lines, Points and Arcs

The primitive elements of the Pooh representation are lines, points and arcs. All algorithms are based on calculations between these elements. The first step in exploring how to represent circuits on the integer grid is to decide how to represent these elements.

Pooh points are circles, represented as octagons. An octagon with radius four represents a unit circle, and points with radii larger than one are scaled accordingly. Figure 8 illustrates example points of various radii. Notice that the intermediate points of an octagon with a half unit radius may not fall on an even grid point, for example the point with radius 1.5. Since these intermediate points are not generated explicitly for any reason except during geometry generation, the fact that they are not on valid grid points is not important.

Lines are restricted to be horizontal, vertical or 45° according to the original criteria of the representation. Lines may be placed anywhere on the grid provided that the line end points are guaranteed to be on valid grid points. The intersection of any two legal lines is always on a valid grid point, with one exception that is

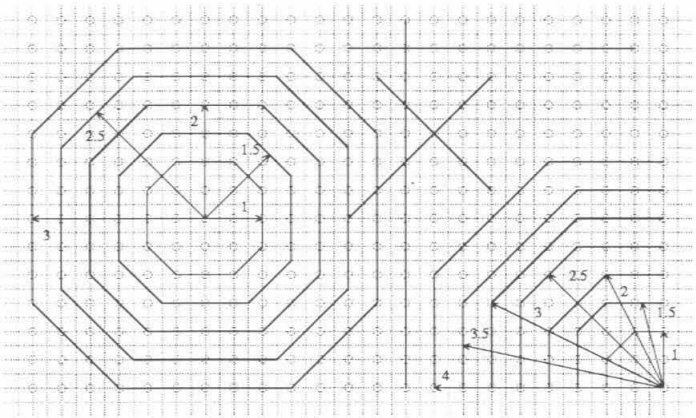


Figure 8: Circles, Lines and Arcs

described later in this paper. There are eight different directions available. The line equations for each of these lines are shown in the following table.

	A	B	C
$\leftarrow$	0	-1	y
$\rightarrow$	0	1	$-y$
$\uparrow$	-1	0	x
$\downarrow$	1	0	$-x$
$\nearrow$	$-\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}(x-y)$
$\searrow$	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}(-x-y)$
$\swarrow$	$\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}(y-x)$
$\nwarrow$	$-\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}(x+y)$

Table 1: The Line Equations

A great simplification results when the line equation $Ax + By + C = 0$ for valid lines is rewritten as $\frac{1}{m}(A'x + B'y + C') = 0$, where the coefficients $A', B' \in \{0, \pm 1\}$, and C' is always an even integer. The values of both the original coefficients and the new coefficients for the three types of lines are shown in table 2.

	A	B	C	m	A'	B'	C'
Horizontal	0	± 1	$\mp y$	1	0	± 1	$\mp y$
Vertical	± 1	0	$\mp x$	1	± 1	0	$\mp x$
$45°$	$\pm\frac{1}{\sqrt{2}}$	$\pm\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}(\mp x \mp y)$	$\sqrt{2}$	± 1	± 1	$\mp x \mp y$

Table 2: Modified Line Equations

Pooh arcs are described by signed radius. Unfortunately the points generated for an arc with a half unit radius are not always on a valid grid point, as illustrated by the points in figure 8, and therefore the end point of the associated line segment is not on a valid grid point. But fortunately, based on the arc radius and the octagon, there is a straight-forward mapping between an octagonal arc and a sequence of connected line segments, as shown in the figure. Notice that each line segment is the nearest segment to the point where both the arc radius distance is preserved and the intersection between this line and its two neighbors is on a valid grid point. In this restricted methodology, arcs are strictly a notational convenience for the designer.

6 Point-Point Calculations

The point-point distance calculation is one of the most crucial comparisons for any system that must cope with spacing in the plane. Even if the system circle approximation is a box, as in manhattan systems, point-point distance calculation must be addressed, as illustrated in figure 9. Some systems address this calculation by mapping the distance into the orthogonal axes [6], as shown in the figure, resulting in significant area inefficiency. The point-point distance calculation is used to compute distances between two points, such as contact-contact or between the endpoints of a path, as shown in figure 10.

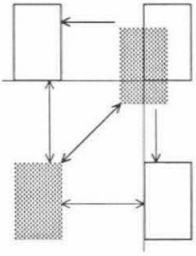


Figure 9: Box Point-Point Distance Calculation

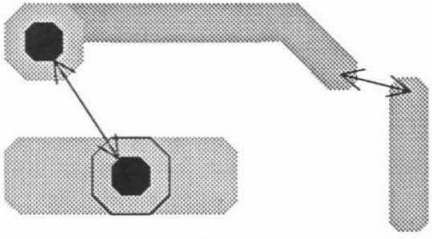


Figure 10: Point-Point Distance Calculation

Simple Euclidean distance, equation (2), involves a square root and is therefore to be avoided. Our task is to convert to integers the calculation of the distance between two points. One possibility is to remove the square root by comparing

the point distance to the GDR distance squared. This approach is valid, however it is susceptible to errors introduced by the octagonal approximation.

An alternative approach is illustrated in Figure 11. Notice that given two points p_1 and p_2, we may construct four lines through p_2, parallel to the four unique sides of the octagon surrounding p_1. Each of these lines, denoted as L_1, L_2, L_3, and L_4, represents one side of an octagon centered at p_1, with a radius equal to the distance between p_1 and the line. The largest of these octagons includes the point p_2 on its boundary.

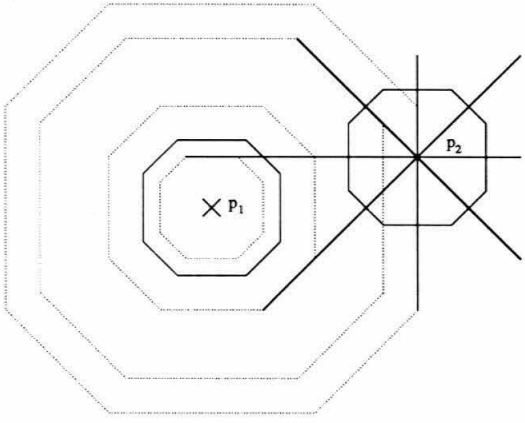


Figure 11: Distance Between Two Points

Given two points and the integer grid, it is always possible to construct an octagon surrounding p_1, with p_2 on its boundary. Since there are only four unique line types in an octagon, at least one of the four lines L_1, L_2, L_3, or L_4 is on this octagon. Since the line segments that comprise this octagon are evenly spaced around p_1, it is not possible to construct a line with a larger distance to p_1 that still intersects p_2. Thus we define the point distance between p_1 and p_2 to be the maximum of the line-point distances, described fully in the next section, between the lines through p_2, L_1, L_2, L_3 and L_4, and the point p_1. This distance is the closest octagon radius to the actual Euclidean point distance between the two points, is always greater than or equal to the actual distance, and is not susceptible to circle approximation errors.

The sign of the line-point distance between these four lines and p_1 is not important, thus we choose four of the eight possible line equation, each with a unique slope. The four line equations of lines that intersect the point p_2 are shown in the following table.

	A	B	C	$D = Ax_1 + By_1 + C$
$\rightarrow$	0	1	$-y_2$	$y_1 - y_2$
$\downarrow$	1	0	$-x_2$	$x_1 - x_2$
$\searrow$	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}(-x_2 - y_2)$	$\frac{1}{\sqrt{2}}(x_1 + y_1 - x_2 - y_2)$
$\swarrow$	$\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}(y_2 - x_2)$	$\frac{1}{\sqrt{2}}(x_1 - y_1 + y_2 - x_2)$

The calculation is:

```
define points_too_close(p1, p2, D): boolean
   I1 ← x1 − x2; I2 ← y1 − y2;
   if |I1| max |I2| > D then points_too_close ← false
   ef ½[|(I1 + I2)| max |(I1 − I2)|]² > D² then
      points_too_close← false
   else points_too_close ← true fi
enddef
```

This function is integer based since both I_1 and I_2 are defined by the even coordinates x and y. This calculation performs fewer operations than the point distance calculation, and the distance is based on the octagon circle approximation, thus circle approximation errors are not present.

7 Line Calculations

There are two important line calculations upon which all Pooh algorithms depend. If these two calculations are integer based, all of the line calculations in Pooh become integer based. These two calculations are: (1) the distance between a point and a line, and (2) the line intersection calculation.

Line-Point Distance Calculation

The distance calculation, equation (3), is $D = LinePoint(L,p)$. There are two important pieces of information in this calculation: the sign of the distance and the actual distance from a point to a line. The sign of $Ax + By + C$ is the same as the sign of $A'x + B'y + C'$, from Table 2. The square of the GDR distance may be stored by the system and compared to the square of the line equation. If a point is less than the GDR distance from a line, the square of the distance from the point to the line is less than the square of the GDR distance. These two equations are:

$$Sign(D) = A'x + B'y + C'$$

$$D^2 = \frac{1}{m^2}(A'x + B'y + C')^2.$$

The number D^2 is guaranteed to be an integer. If the line is horizontal or vertical then $m = 1$, $1/m^2 = 1$, $A', B' \in \{0, \pm 1\}$, x and y are integer, therefore D^2 is an integer. If the line is 45°, then $m = \sqrt{2}$, $1/m^2 = 1/2$, and the 45° distance equation may be rewritten as:

$$D^2 = 1/2 \times \left[2 \times \left(\pm 1(x/2) \pm 1(y/2) + (\mp x \mp y)/2\right)\right]^2$$
$$= 2[\pm 1(x/2) \pm 1(y/2) + (\mp x \mp y)/2]^2.$$

Since both x and y are even, the bracketed computation is an integer, making the entire computation an integer one.

Notice that the distance calculation is not susceptible to errors introduced from the octagonal approximation. Since all valid line types are parallel to an octagon edge, the distance is conservative as described in the previous section.

Line Intersection Calculation

The line intersection calculation, equation (1), determines the point at which two lines intersect, and is used extensively, often in a simplified form. If the determinant is zero, the two lines are parallel. This condition can be detected when the determinant is calculated and processing of the two lines terminates. For two lines that are not parallel, the computation proceeds as follows.

If the two lines in question are vertical and horizontal, then the line coefficients are 0 or ± 1, i.e. $A_1, A_2, B_1, B_2 \in \{0, \pm 1\}$, thus det $\in \{0, \pm 1\}$. If the determinant is 0, the lines are parallel, and therefore uninteresting. Otherwise the only information of interest is the sign of "det". Multiplying by ± 1 is equivalent to dividing, thus the equation may be rewritten as:

$$p_i = \det \times \left((B_2C_1 - B_1C_2),\ (A_1C_2 - A_2C_1)\right).$$

If both lines are 45°, the determinant is either 0 or ± 1, as is always true for parallel and perpendicular lines. If the determinant is non-zero, then we may multiply by "det", rather than dividing. The form of the intersection between these two lines is:

$$p_i = \det \times \left[\pm \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2}}(\pm x \pm y)\right) - \pm \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2}}(\pm x \pm y)\right),\right.$$
$$\left. \pm \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2}}(\pm x \pm y)\right) - \pm \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2}}(\pm x \pm y)\right)\right]$$
$$p_i = \det \times \frac{1}{2} \times (B'_2C'_1 - B'_1C'_2,\ A'_1C'_2 - A'_2C'_1).$$

The point p_i is an integer because: (a) the coefficients A' and B' are ± 1, and the C' coefficients are combinations of the even x and y coordinates, (b) the component det is ± 1, and (c) the factor $1/2$ applied to even integers produces integers.

If one line is 45°, and the other is vertical, then the calculation is of the form:

$$\det = \pm\frac{1}{\sqrt{2}}(\pm 1) - \left(\pm\frac{1}{\sqrt{2}}\right) \times 0 = \pm\frac{1}{\sqrt{2}}$$

$$p_i = \left[\left(B_2\left(\frac{\pm x_1 \pm y_1}{\sqrt{2}}\right) - \pm\frac{1}{\sqrt{2}}C_2\right) / \left(\pm\frac{1}{\sqrt{2}}\right),\right.$$
$$\left.\left(\pm\frac{1}{\sqrt{2}}C_2 - A_2\left(\frac{\pm x_1 \pm y_1}{\sqrt{2}}\right) / \left(\pm\frac{1}{\sqrt{2}}\right)\right)\right]$$

$$p_i = \sqrt{2}\left[\frac{1}{\sqrt{2}}(B_2C_1' - B_1'C_2,\ A_1'C_2 - A_2C_1')\right]$$

$$p_i = (B_2C_1' - B_1'C_2, A_1'C_2 - A_2C_1').$$

The intersection between 45° and horizontal lines works in an identical fashion. In general, the intersection between two lines may be calculated as:

$$\det' = B_1'A_2' - A_1'B_2'$$

$$p_i = \det' \times \left(\frac{1}{m_1^2} \max \frac{1}{m_2^2}\right) \times (B_2'C_1' - B_1'C_2', A_1'C_2' - A_2'C_1').$$

Angle Calculations

The remaining algorithm for lines is detecting an illegal angle between connecting lines. Since there are only eight possible line types, the simplest solution is to calculate the angle between the eight types of lines once, and perform a table lookup based on the types of the two lines in question. In practice, we require important intersections, i.e. those forming transistors, to be perpendicular.

8 A Problem

There are always problems in any system that attempts to restrict its domain. Usually these problems are not insurmountable, but they must be addressed. For example, in a strictly manhattan system that enforces GDRs, the system must decide what to do about the non-manhattan distances of corner to corner spacings. In the octagon representation, there is an inherent problem with the interactions between valid line segments.

The problem is that two perfectly legal 45° lines may intersect at an illegal grid point. Figure 12(a) illustrates two line segments A and B that intersect at an odd grid point. Since all valid points are even grid points, shown as circles in the figure, this situation must be detected and prevented. Figure 12(b) illustrates two legal possibilities B' and B'' where the segment B (or the current segment) is automatically moved by the system depending on which situation is GDR correct. Unfortunately on occasion the segment B is constrained by other structures and thus cannot be moved. Figure 12(c) illustrates a corresponding legal situation, where segment B is automatically mapped into five connected lines, B_1, through B_5. Notice that (b) and (c) may introduce GDR violations not present in (a). Such an error is caught by the analysis algorithms and is identical to any other GDR violation. Of course non-interacting GDR correct lines may cross freely since the

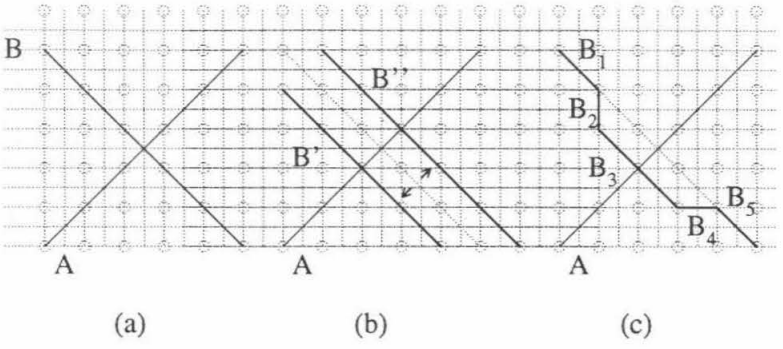


Figure 12: Invalid and Valid 45° Segments

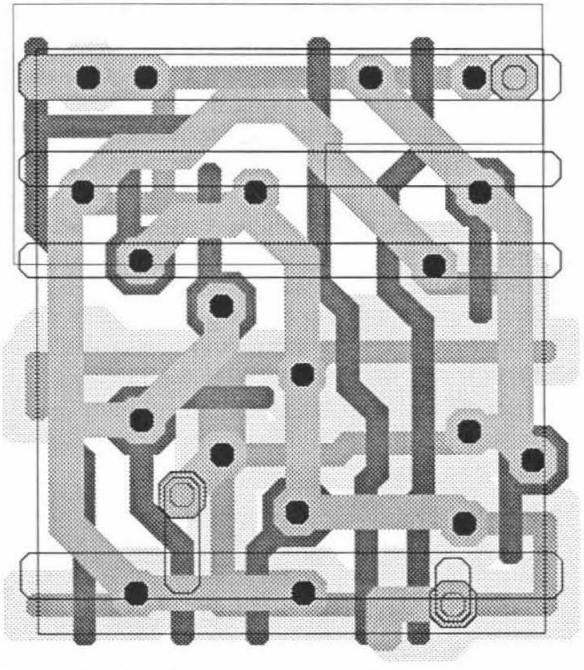


Figure 13: The Integer version of the Bit Serial Multiplier Cell

point of intersection is not represented explicitly and therefore is not required to be on the grid.

9 Conclusions

We have presented a simple hierarchical representation for circuits whose algorithms for GDR enforcement are simplified greatly by the restricted line styles

and integer grid. The effective use of orthogonal and 45° lines not only simplifies the circuit description but reduces the design area without any computational cost. In fact this approach requires less computation than typical manhattan systems. This methodology not only allows the detection of errors between transistors and wires, but guarantees that a composition of two or more circuit descriptions is correct, given that the individual circuits are correct.

All of the synthesis algorithms depend on the line equations and rely heavily on the line intersection calculation. Most of the path construction calculations are simplifications of the line intersection. The angle between lines may be calculated once, since there are a small number of unique legal line slopes, and looked up for a particular interaction.

The analysis algorithms use the line-point distance and the point-point distance as the basis for all GDR violation detection techniques, and do not require a square root or a division. A simplification of the line intersection calculation allows Pooh to detect whether a point is between the two end points of a line segment. Thus all analysis algorithms are integer based.

By imposing some restrictions on the angle of valid lines, the Pooh representation and its associated algorithms are both simplified and integer based. Most of the multiplications use the coefficients 0, ± 1, and 2. None of these numbers actually require a real multiplication, instead they may be implemented as shifts and sign changes to increase the numerical efficiency even further.

Cell composition uses the line equations to represent all cell sides. The integer version of the composition restricts the valid sides to vertical, horizontal or 45°. In order to preserve the integer grid, cell rotation and mirroring is restricted to multiples of ninety degrees. The line intersection calculation is used during composition in the construction of the cell interface. Finally the analysis algorithms are used to detect GDR violations between composed cells. Thus the entire composition approach is also integer based.

Figure 13 illustrates a bit serial multiplier cell using strictly orthogonal and 45° lines. This integer version of the cell is the same size as the original cell designed with geometry at arbitrary angles.

Acknowledgments

We would like to acknowledge Dr. Ivan Sutherland whose polygon package ideas inspired the work. Special appreciation to members of "the group" in the Caltech Computer Science Lab for their support, ideas and critiques. Thanks to Dick Lyon for his careful comments, and to John Wawrzynek for his help.

This work was supported by the System Development Foundation.

Bibliography

[1] Baker, C.M., and Terman, C.,
Tools for Verifying Integrated Circuit Designs,
Lambda Magazine 4th Quarter:22-30, 1980.

[2] Mead, C., and Conway, L., **Introduction to VLSI Systems,**
Addison-Wesley, Reading, Massachusetts, 1980.

[3] Mead, C., **The Wolery**, Technical Report # 5113:TR:84, California Institute of Technology, January, 1984.

[4] Mead, C.A, Wawrzynek, J., **A Discipline for cMOS Design: An Architecture for Sound Synthesis**, Proceedings of the Chapel Hill Conference on VLSI, 1985.

[5] Maley, F.M., **Compaction with Automatic Jog Introduction**, Proceedings of 1985 Chapel Hill Conference on VLSI, pages 261-283, 1985.

[6] Ousterhout, J.K., Hamachi, G.T., Mayo, R.N., Scott, W.S., Taylor, G.S., **Magic: A VLSI Layout System**, Proceedings of 21st D.A. Conference, pages 152-159, June, 1984.

[7] Seiler, L.D., **A Hardware Assisted Methodology for VLSI Design Rule Checking**, PhD Thesis, Massachusetts Institute of Technology, 1985.

[8] Sutherland, I.E., **The Polygon Package**, Technical Report # 1438, California Institute of Technology, February, 1978.

[9] Whitney, T., and Mead, C.A., **Pooh: A Uniform Representation for Circuit Level Designs**, Proceedings of International Conference on VLSI, Trondheim, Norway, pages 401-411, August, 1983.

[10] Whitney, T.E., **Hierarchical Composition of VLSI Circuits**, PhD Thesis, Computer Science Dept., California Institute of Technology, 1985.

[11] Williams, J., **Sticks—A New Approach to LSI Design**, Master's Thesis, Massachusetts Institute of Technology, June, 1977.